Intel® Pentium® M Processor

Specification Update

October 2006

Notice: The Intel® Pentium® M processor may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are documented in this Specification Update.

Document Number: 252665-026

Contents

Revision History

Revision Number	Description	Date
001	Initial Release	March 2003
002	• Added Erratum Y15 • Clarified Status description for all Erratum	April 2003
003	• Updated Pentium M processor Identification Table	June 2003
004	• Added Erratum Y16, Y17	July 2003
005	• Added Erratum Y18	September 2003
006	• Added Erratum Y19 - Y21	November 2003
007	• Updated Erratum Y20 • Added Erratum Y22	December 2003
008	• Added Erratum Y23 and Y24 • Added Specification Clarification Y1 • Updated Processor Identification table	April 2004
009	• Added Erratum Y25 and Y26	May 2004
010	• Added Erratum Y27, Y28, Y29 and Y30	October 2004
011	• Added Erratum Y31, Y32 and Y33	November 2004
012	• Added Erratum Y34 and Y35	December 2004
013	• Updated Summary Tables of Changes • Added Erratum Y36	February 2005
014	• Added Erratum Y37 • Added Specification Clarification Y1	May 2005
015	• Updated Summary Tables of Changes • Removed Erratum Y28 – Y30 (which were duplicates of Y4 – Y6). • Added Erratum Y38 – Y40	June 2005
016	• Updated Affected and Related Documents Tables	September 2005
017	• Added Erratum Y41	November 2005
018	• Added Erratum Y42 and Y43	December 2005
019	• Added Erratum Y44 – Y48 • Removed Specification Clarification Y1; refer to Section 18.8 of the *IA-32 Intel® Architecture Software Developer's Manual, Volume 3B* for detailed Time Stamp Counter information.	January 2006
020	• Added Erratum Y49	February 2006
021	• Update processors code section • Updated Erratum Y7 • Updated Erratum Y34	March 2006

022	• Updated processor name for AF in processor code • Added Erratum Y50	April 2006
023	• Updated Errata Y27 • Updated Errata Y43 • Added new Errata Y51 • Added SDM Changes link in 'Documentation Changes' section	May 2006
024	• Updated Errata Y12 • Updated Errata Y38 – Workaround updated. • Added new Errata Y52 • Added new Errata Y53	July 2006
025	• Updated Errata Y19 • Added new Errata Y54, Y55, Y56	August 2006
026	• Updated names of Software Developer's Manuals • Updated Summary Table of Changes (Product prefix table) • Add new Errata Y57	October 2006

Preface

This document is an update to the specifications contained in the documents listed in the following Affected Documents/Related Documents table. It is a compilation of device and document errata and specification clarifications and changes, and is intended for hardware system manufacturers and for software developers of applications, operating system, and tools.

Information types defined in the Nomenclature section of this document are consolidated into this update document and are no longer published in other documents. This document may also contain information that has not been previously published.

Affected Documents

Document Title	Document Number
Intel® Pentium® M Processor Datasheet	252612-003

Related Documents

Document Title	Document Number
Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 1: Basic Architecture	253665
Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2A: Instruction Set Reference, A-M	253666
Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2B: Instruction Set Reference, N-Z	253667
Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A: System Programming Guide	253668
Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B: System Programming Guide	253669
Intel® 64 and IA-32 Intel® Architecture Optimization Reference Manual	248966

Nomenclature

S-Spec Number is a five-digit code used to identify products. Products are differentiated by their unique characteristics, e.g., core speed, L2 cache size, package type, etc. as described in the processor identification information table. Care should be taken to read all notes associated with each S-Spec number.

Errata are design defects or errors. Errata may cause the Intel® Pentium® M processor's behavior to deviate from published specifications. Hardware and software, designed to be used with any given processor, must assume that all errata documented for that processor are present on all devices unless otherwise noted.

Documentation Changes include typos, errors, or omissions from the current published specifications. These changes will be incorporated in the next release of the specifications.

Specification Clarifications describe a specification in greater detail or further highlight a specification's impact to a complex design situation. These clarifications will be incorporated in the next release of the specifications.

Specification Changes are modifications to the current published specifications for the Pentium M processor. These changes will be incorporated in the next release of the specifications.

Summary Tables of Changes

The following table indicates the Errata, Documentation Changes, Specification Clarifications, or Specification Changes that apply to Pentium M processors. Intel intends to fix some of the errata in a future stepping of the component and to account for the other outstanding issues through documentation or specification changes as noted. This table uses the following notations:

Codes Used in Summary Table

Stepping

X:	Erratum, Specification Change or Clarification that applies to this stepping.
(No mark) or (Blank Box):	This erratum is fixed in listed stepping or specification change does not apply to listed stepping.

Status

Doc:	Document change or update that will be implemented.
PlanFix:	This erratum may be fixed in a future of the product.
Fixed:	This erratum has been previously fixed.
NoFix:	There are no plans to fix this erratum.
Shaded:	This item is either new or modified from the previous version of the document.

Each Specification Update item is prefixed with a capital letter to distinguish the product. The key below details the letters that are used in Intel's microprocessor Specification Updates:

A = Dual-Core Intel® Xeon® processor 7000 sequence

B = Mobile Intel® Pentium® II processor

C = Intel® Celeron® processor

D =Dual-Core Intel® Xeon® processor 2.80 GHz

E = Intel® Pentium® III processor

F = Intel® Pentium® processor extreme edition and Intel® Pentium® D processor

G = Intel® Pentium® III Xeon™ processor

H = Mobile Intel® Celeron® processor at 466 MHz, 433 MHz, 400 MHz, 366 MHz, 333 MHz, 300 MHz, and 266 MHz

J = 64-bit Intel® Xeon™ processor MP with 1-MB L2 cache

K = Mobile Intel® Pentium® III Processor – M

L = Intel® Celeron® D processor

M =Mobile Intel® Celeron® processor

N = Intel® Pentium® 4 processor

O = Intel® Xeon™ processor MP

P = Intel® Xeon™ processor

Q = Mobile Intel® Pentium® 4 processor supporting Hyper-Threading Technology† on 90-nm technology process

R = Intel® Pentium® 4 processor on 90 nm process

S = 64-bit Intel® Xeon™ processor with 800 MHz system bus (1-MB and 2-MB L2 cache versions)

T = Mobile Intel® Pentium® 4 processor – M

U = 64-bit Intel® Xeon™ processor MP with up to 8-MB L3 cache

V = Mobile Intel® Celeron® processor on .13 Micron process in Micro-FCPGA package

W = Intel® Celeron® M processor

X = Intel® Pentium® M processor on 90 nm process with 2-MB L2 cache

Y = Intel® Pentium® M processor

Z = Mobile Intel® Pentium® 4 processor with 533 MHz system bus

AA = Intel® Pentium® processor Extreme Edition and Intel® Pentium® D processor on 65nm process

AB = Intel® Pentium® 4 processor on 65 nm process

AC = Intel(R) Celeron(R) processor in 478 Pin Package

AD = Intel(R) Celeron(R) D processor on 65nm process

AE = Intel® Core™ Duo Processor and Intel® Core™ Solo processor on 65nm process

AF = Dual-Core Intel® Xeon® processor LV

AG = Dual-Core Intel® Xeon® Processor 5100 Δ Series
AH = Intel® Core™2 Duo mobile processor
AI = Intel® Core™2 Extreme Processor X6800Δ and Intel® Core™2 Duo Desktop Processor E6000Δ Sequence
AL = Dual-Core Intel® Xeon® Processor 7100 Δ Series

Note: The Specification Updates for the Pentium® processor, Pentium® Pro processor, and other Intel products do not use this convention.

NO.	Stepping	Plans	ERRATA
	B1		
Y1	X	NoFix	Performance Monitoring Event That Counts Intel® Thermal Monitor 2 Transitions (59h) Is Not Accurate

NO.	Stepping	Plans	ERRATA
	B1		
Y2	X	NoFix	Performance Monitoring Event That Counts the Number of Instructions Decoded (D0h) Is Not Accurate
Y3	X	NoFix	RDTSC Instruction May Report the Wrong Time-Stamp Counter Value
Y4	X	NoFix	Code Segment Limit Violation May Occur on 4-Gbyte Limit Check
Y5	X	NoFix	FST Instruction with Numeric and Null Segment Exceptions May Cause General Protection Faults to Be Missed and FP Linear Address (FLA) Mismatch
Y6	X	NoFix	Code Segment (CS) Is Wrong on SMM Handler When SMBASE Is Not Aligned
Y7	X	NoFix	A Locked Data Access that Spans Across Two Pages May Cause the System to Hang
Y8	X	NoFix	Processor Can Enter a Livelock Condition under Certain Conditions When FP Exception Is Pending
Y9	X	NoFix	Write Cycle of Write Combining Memory Type Does Not Self Snoop
Y10	X	NoFix	Performance Monitoring Event That Counts Floating Point Computational Exceptions (11h) Is Not Accurate.
Y11	X	NoFix	Inconsistent Reporting of Data Breakpoints on FP (MMX™ technology) Loads
Y12	X	No Fix	Code breakpoint May Be Taken after POP SS Instruction if it is followed by an Instruction that Faults
Y13	X	NoFix	SysEnter and SysExit Instructions May Write Incorrect Requestor Privilege Level (RPL) in the FP Code Segment Selector (FCS)
Y14	X	NoFix	Memory Aliasing with Inconsistent A and D Bits May Cause Processor Deadlock
Y15	X	NoFix	RDMSR or WRMSR to Invalid MSR Address May Not Cause GP Fault
Y16	X	NoFix	FP Tag Word Corruption
Y17	X	NoFix	Unable to Disable Reads/Writes to Performance Monitoring Related MSRs
Y18	X	NoFix	Move to Control Register Instruction May Generate a Breakpoint Report
Y19	X	NoFix	REP MOVS/STOS Executing with Fast Strings Enabled and Crossing Page Boundaries with Inconsistent Memory Types may use an Incorrect Data Size or Lead to Memory-Ordering Violations
Y20	X	NoFix	The FXSAVE, STOS, or MOVS Instruction May Cause a Store Ordering Violation When Data Crosses a Page with a UC Memory Type
Y21	X	NoFix	Machine Check Exception May Occur Due to Improper Line Eviction in the IFU
Y22	X	NoFix	POPF and POPFD Instructions That Set the Trap Flag Bit May Cause Unpredictable Processor Behavior
Y23	X	NoFix	Performance Event Counter Returns Incorrect Value on L2_LINES_IN Event
Y24	X	NoFix	VM Bit Will Be Cleared on a Double Fault Handler
Y25	X	NoFix	Code Fetch Matching Disabled Debug Register May Cause Debug Exception
Y26	X	NoFix	Upper Four PAT Entries Not Usable with Mode B or Mode C Paging
Y27	X	NoFix	SSE/SSE2 Streaming Store Resulting in a Self-Modifying Code (SMC) Event May Cause Unexpected Behavior

NO.	Stepping	Plans	ERRATA
	B1		
Y28			Removed, see Erratum Y4
Y29			Removed, see Erratum Y5
Y30			Removed, see Erratum Y6
Y31	X	NoFix	Page with PAT (Page Attribute Table) Set to USWC (Uncacheable Speculative Write Combine) While Associated MTRR (Memory Type Range Register) Is UC (Uncacheable) May Consolidate to UC
Y32	X	NoFix	Under Certain Conditions LTR (Load Task Register) Instruction May Result in System Hang
Y33	X	NoFix	Loading from Memory Type USWC (Uncacheable Speculative Write Combine) May Get Its Data Internally Forwarded from a Previous Pending Store
Y34	X	NoFix	FPU Operand Pointer May Not be Cleared Following FINIT/FNINIT
Y35	X	NoFix	FSTP (Floating Point Store) Instruction Under Certain Conditions May Result In Erroneously Setting a Valid Bit on an FP (Floating Point) Stack Register
Y36	X	NoFix	Snoops during the Execution of a HLT (Halt) Instruction May Lead to Unpredictable System Behavior
Y37	X	NoFix	Invalid Entries in Page-Directory-Pointer-Table Register (PDPTR) May Cause General Protection (#GP) Exception if the Reserved Bits are Set to One
Y38	X	No Fix	INIT does not clear global entries in the TLB
Y39	X	NoFix	Use of Memory Aliasing with Inconsistent Memory Type May Cause System Hang
Y40	X	NoFix	Machine Check Exception May Occur When Interleaving Code between Different Memory Types
Y41	X	NoFix	Split I/O Writes Adjacent to Retry of APIC End of Interrupt (EOI) Request May Cause Livelock Condition
Y42	X	No Fix	General Protection (#GP) Fault May Not Be Signaled On Data Segment Limit Violation above 4-G Limit
Y43	X	No Fix	DR3 Address Match on MOVD/MOVQ/MOVNTQ Memory Store Instruction May Incorrectly Increment Performance Monitoring Count for Saturating SIMD Instructions Retired (Event CFH)
Y44	X	No Fix	Processor INIT# Will Cause a System Hang if Triggered During an NMI Interrupt Routine Performed During Shutdown
Y45	X	No Fix	Certain Performance Monitoring Counters Related to Bus, L2 Cache and Power Management are Inaccurate
Y46	X	No Fix	CS Limit Violation on RSM May be Serviced before Higher Priority Interrupts/Exceptions
Y47	X	No Fix	A Write to an APIC Register Sometimes May Appear to Have Not Occurred
Y48	X	No Fix	The Processor May Report a #TS Instead of a #GP Fault
Y49	X	No Fix	Writing the Local Vector Table (LVT) when an interrupt is pending may cause an unexpected interrupt
Y50	X	No Fix	Using 2M/4M Pages When A20M# is Asserted May Result in Incorrect Address Translations

NO.	Stepping B1	Plans	ERRATA
Y51	X	No Fix	Premature Execution of a Load Operation Prior to Exception Handler Invocation
Y52	X	No Fix	Incorrect Address Computed For Last Byte of FXSAVE/FXRSTOR Image Leads to Partial Memory Update
Y53	X	No Fix	Values for LBR/BTS/BTM will be Incorrect after an Exit from SMM
Y54	X	No Fix	FP Inexact-Result Exception Flag May Not Be Set
Y55	X	No Fix	MOV To/From Debug Registers Causes Debug Exception
Y56	X	No Fix	SYSENTER/SYSEXIT Instructions Can Implicitly Load “Null Segment Selector” to SS and CS Registers
Y57	X	No Fix	The BS Flag in DR6 May be Set for Non-Single-Step #DB Exception

Number	SPECIFICATION CHANGE
	There are no Specification Changes in this Specification Update revision.

Number	SPECIFICATION CLARIFICATIONS
	There are no Specification Clarifications in this Specification Update revision.

Number	DOCUMENTATION CHANGES
	There are no Documentation Changes in this Specification Update revision.

Identification Information

The Pentium M processor can be identified by the following values:

Family[1]	Model[2]	Brand ID[3]
0110	1001	00010110

NOTES:
1. The Family corresponds to bits [11:8] of the EDX register after Reset, bits [11:8] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register.
2. The Model corresponds to bits [7:4] of the EDX register after Reset, bits [7:4] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register.
3. The Brand ID corresponds to bits [7:0] of the EBX register after the CPUID instruction is executed with a1 in the EAX register.

Table 1. Intel Pentium M Processor Identification

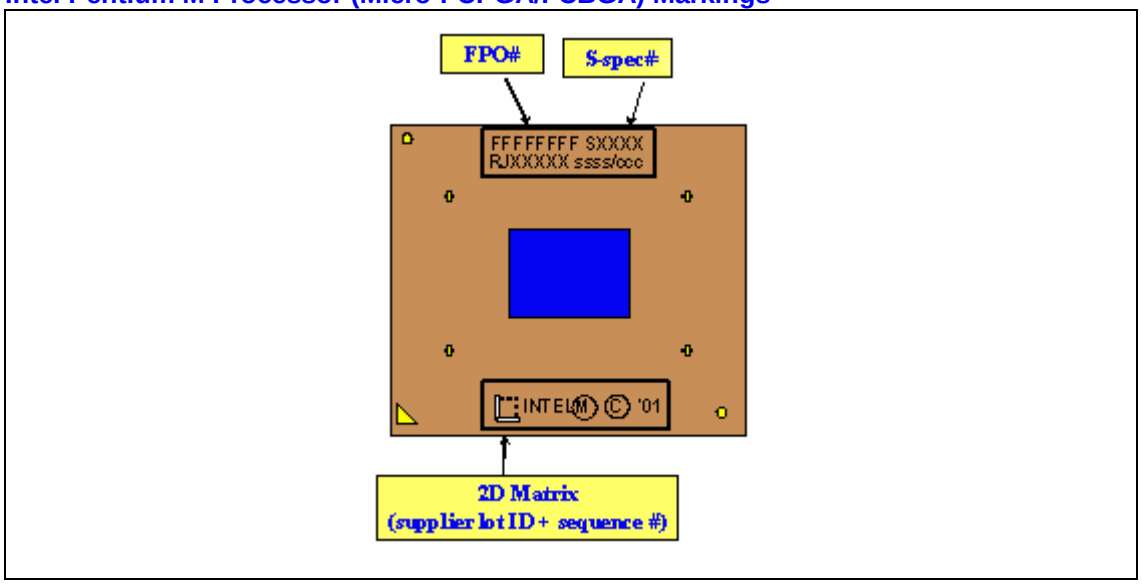
S-Spec/ QDF	Product Stepping	CPUID	Core Speed		Bus Frequency	Package	Notes
			Highest Frequency Mode (HFM)	Lowest Frequency Mode (LFM)			
SL6N4	B-1	0695h	1.30 GHz	600 MHz	400 MHz	Micro-FCPGA	2,6
SL6F8	B-1	0695h	1.40 GHz	600 MHz	400 MHz	Micro-FCPGA	2,5
SL6F9	B-1	0695h	1.50 GHz	600 MHz	400 MHz	Micro-FCPGA	2,5
SL6FA	B-1	0695h	1.60 GHz	600 MHz	400 MHz	Micro-FCPGA	2,5
SL6N5	B-1	0695h	1.70 GHz	600 MHz	400 MHz	Micro-FCPGA	2,5
SL6N8	B-1	0695h	1.30 GHz	600 MHz	400 MHz	Micro-FCBGA	2,6
SL6F5	B-1	0695h	1.40 GHz	600 MHz	400 MHz	Micro-FCBGA	2,5
SL6F6	B-1	0695h	1.50 GHz	600 MHz	400 MHz	Micro-FCBGA	2,5
SL6F7	B-1	0695h	1.60 GHz	600 MHz	400 MHz	Micro-FCBGA	2,5
SL6N9	B-1	0695h	1.70 GHz	600 MHz	400 MHz	Micro-FCBGA	2,5
SL6NC	B-1	0695h	1.10 GHz	600 MHz	400 MHz	Micro-FCBGA	1,2
SL6NB	B-1	0695h	1.20 GHz	600 MHz	400 MHz	Micro-FCBGA	1,2
SL6NA	B-1	0695h	1.30 GHz	600 MHz	400 MHz	Micro-FCBGA	1, 2
SL6NJ	B-1	0695h	900 MHz	600 MHz	400 MHz	Micro-FCBGA	3,4
SL6NH	B-1	0695h	1.00 GHz	600 MHz	400 MHz	Micro-FCBGA	3,4
SL6P4	B-1	0695h	1.10 GHz	600 MHz	400 MHz	Micro-FCBGA	3,4

NOTES:
1. VID[5:0] = 100001; VCC_CORE = 1.180 V for Highest Frequency Mode (HFM).
2. VID[5:0] = 101111; VCC_CORE = 0.956 V for Lowest Frequency Mode (LFM).
3. VID[5:0] = 101100; VCC_CORE = 1.004 V for Highest Frequency Mode (HFM).

4. VID[5:0] = 110110; VCC_CORE = 0.844 V for Lowest Frequency Mode (LFM).
5. VID[5:0] = 001110; VCC_CORE = 1.484 V for Highest Frequency Mode (HFM).
6. VID[5:0] = 010100; VCC_CORE = 1.388 V for Highest Frequency Mode (HFM).

Component Marking Information

Figure 1. Intel Pentium M Processor (Micro-FCPGA/FCBGA) Markings



§

Errata

Y1. Performance Monitoring Event That Counts Intel® Thermal Monitor 2 Transitions (59h) Is Not Accurate

Problem: The performance monitoring event that counts Intel Thermal Monitor 2 (Enhanced Intel SpeedStep® technology based) transitions may have inaccurate results.

Implication: There is no functional impact of this erratum. However this Performance Monitoring Event should not be used when accurate performance monitoring is required.

Workaround: None.

Status: For the steppings affected, see the *Summary of Tables of Changes*.

Y2. Performance Monitoring Event that Counts the Number of Instructions Decoded (D0h) Is Not Accurate

Problem: The performance-monitoring event that counts the number of instructions decoded may have inaccurate results.

Implication: There is no functional impact of this erratum. However the results/counts from this Performance Monitoring Event should not be considered as being accurate

Workaround: None.

Status: For the steppings affected, see the *Summary of Tables of Changes*.

Y3. RDTSC Instruction May Report the Wrong Time-stamp Counter Value

Problem: The Time-stamp Counter is a 64-bit counter that is read in two 32-bit chunks. The counter incorrectly advances and therefore the two chunks may go out of synchronization causing the Read Time-stamp Counter (RDTSC) instruction to report the wrong time-stamp counter value

Implication: This erratum may cause software to see the wrong representation of processor time and may result in unpredictable software operation.

Workaround: It is possible for BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary of Tables of Changes*.

Y4. Code Segment Limit Violation May Occur on 4 Gigabyte Limit Check

Problem: Code Segment limit violation may occur on 4 Gigabyte limit check when the code stream wraps around in a way that one instruction ends at the last byte of the segment and the next instruction begins at 0x0.

Implication: This is a rare condition that may result in a system hang. Intel has not observed this erratum with any commercially available software, or system.

Workaround: Avoid code that wraps around segment limit.

Status: For the steppings affected, see the *Summary of Tables of Changes*.

Y5. FST Instruction with Numeric and Null Segment Exceptions May Cause General Protection Faults to Be Missed and FP Linear Address (FLA) Mismatch

Problem: FST instruction combined with numeric and null segment exceptions may cause General Protection Faults to be missed and FP Linear Address (FLA) mismatch.

Implication: This is a rare condition that may result in a system hang. Intel has not observed this erratum with any commercially available software, or system.

Workaround: None.

Status: For the steppings affected, see the *Summary of Tables of Changes*.

Y6. Code Segment (CS) Is Wrong on SMM Handler When SMBASE Is Not Aligned

Problem: With SMBASE being relocated to a non-aligned address, during SMM entry the CS can be improperly updated which can lead to an incorrect SMM handler.

Implication: This is a rare condition that may result in a system hang. Intel has not observed this erratum with any commercially available software, or system.

Workaround: Align SMBASE to 32 KB.

Status: For the steppings affected, see the *Summary of Tables of Changes*.

Y7. A Locked Data Access that Spans Across Two Pages May Cause the System to Hang

Problem: An instruction with lock data access that spans across two pages may, given some rare internal conditions, hang the system.

Implication: When this erratum occurs, the system may hang. Intel has not observed this erratum with any commercially available software or system.

Workaround: A locked data access should always be aligned.

Status: For the steppings affected, see the *Summary of Tables of Changes*.

Y8. Processor Can Enter a Livelock Condition under Certain Conditions When FP Exception Is Pending

Problem: Processor clock modulation may be controlled via a processor register (IA32_THERM_CONTROL) or via the STPCLK# signal. While the Processor clock is constantly being actively modulated at 12.5% and 25% duty cycles and there is a pending unmasked FP exception (ES pending), if you attempt a FP load (or MMX™ technology MOV instruction) and the load has an longer than typical latency the processor can enter a livelock.

Implication: When this erratum occurs, the processor will enter a livelock condition. Intel has not observed this erratum with any commercially available software or system.

Workaround: None.

Status: For the steppings affected, see the *Summary of Tables of Changes.*

Y9. Write Cycle of Write Combining Memory Type Does Not Self Snoop

Problem: Write cycles of WC memory type do not self-snoop. This may result in data inconsistency – if the addresses of the WC data are aliased to WB memory type memory, which has been cached. In such a case, the internal caches will not be updated with the WC data sent on the system bus.

Implication: This condition may result in a data inconsistency. Intel has not observed this erratum with any commercially available software, system, nor components.

Workaround: Software should detect via the self-snoop bit in the CPUID features flags if the processor supports a self-snooping capability. Software should perform explicit memory management/flushing for aliased memory ranges on processors that do not self-snoop.

Status: For the steppings affected, see the *Summary of Tables of Changes.*

Y10. Performance Monitoring Event that Counts Floating Point Computational Exceptions (11h) Is Not Accurate

Problem: Performance monitoring event that counts Floating Point Compare exceptions may have inaccurate results.

Implication: There is no functional impact of this erratum. However this Performance Monitoring Event should not be used when accurate performance monitoring is required.

Workaround: None.

Status: For the steppings affected, see the *Summary of Tables of Changes.*

Y11. Inconsistent Reporting of Data Breakpoints on FP (MMX™ Technology) Loads

Problem: The reporting of data breakpoints on either FP or MMX technology loads is dependent upon the code faulting behavior prior to the execution of the load. If there is a fault pending prior to the execution of the load and FP exceptions are enabled there is a chance that data breakpoint on successive FP/MMX technology Loads may be reported twice.

Implication: Software debuggers should be aware of this possibility. There should be no implications to software operated outside of a debug environment.

Workaround: None.

Status: For the steppings affected, see the *Summary of Tables of Changes*.

Y12. Code Breakpoint May Be Taken after POP SS Instruction If It Is followed by an Instruction That Faults

Problem: A POP SS instruction should inhibit all interrupts including Code Breakpoints until after execution of the following instruction. This allows sequential execution of POP SS and MOV eSP, eBP instructions without having an invalid stack during interrupt handling. However, a code breakpoint may be taken after POP SS if it is followed by an instruction that faults, this results in a code breakpoint being reported on an unexpected instruction boundary since both instructions should be atomic.

Implication: This can result in a mismatched Stack Segment and SP. Intel has not observed this erratum with any commercially available software, or system.

Workaround: As recommended in the *IA32 Intel® Architecture Software Developer's Manual*, the use "POP SS" in conjunction with "MOV eSP, eBP" will avoid the failure since the "MOV" will not fault.

Status: For the steppings affected, see the *Summary of Tables of Changes*.

Y13. SysEnter and SysExit Instructions May Write Incorrect Requestor Privilege Level (RPL) in the FP Code Segment Selector (FCS)

Problem: SysEnter and SysExit instructions may write incorrect RPL in the FP Code Segment selector (FCS). As a result of this, the RPL field in FCS may be corrupted.

Implication: This is a rare condition that may result in a system hang. Intel has not observed this erratum with any commercially available software, or system.

Workaround: It is possible for BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary of Tables of Changes*.

Y14. Memory Aliasing with Inconsistent A and D Bits May Cause Processor Deadlock

Problem: In the event that software implements memory aliasing by having two Page Directory Entries (PDEs) point to a common Page Table Entry (PTE) and the Accessed and Dirty bits for the two PDEs are allowed to become inconsistent the processor may become deadlocked.

Implication: This erratum has not been observed with commercially available software.

Workaround: Software that needs to implement memory aliasing in this way should manage the consistency of the Accessed and Dirty bits.

Status: For the steppings affected, see the *Summary of Tables of Changes*.

Y15. RDMSR or WRMSR to Invalid MSR Address May Not Cause GP Fault

Problem: The RDMSR and WRMSR instructions allow reading or writing of MSRs (Model Specific Registers) based on the index number placed in ECX. The processor should reject access to any reserved or unimplemented MSRs by generating #GP(0). However, there are some invalid MSR addressers for which the processor will not generate #GP(0). This erratum has not been observed with commercially available software.

Implication: For RDMSR, undefined values will be read into EDX:EAX. For WRMSR, undefined processor behavior may result.

Workaround: Do not use invalid MSR addresses with RDMSR or WRMSR.

Status: For the steppings affected, see the *Summary of Tables of Changes*.

Y16. FP Tag Word Corruption

Problem: In some rare cases, fault information generated as the result of instruction execution may be incorrect. The result is an incorrect FP stack entry.

Implication: This erratum may result in corruption of the FP Tag Word in a way that a non-valid entry in the FP Stack may become valid. The software is not expected to read a non-valid entry. If the software attempts to use the stack entry (which is expected to be empty) the result may be an erroneous “Stack overflow”.

Workaround: Do not disable SSE/SSE2 in control register CR4 and avoid Code Segment Limit violation.

Status: For the steppings affected, see the *Summary of Tables of Changes*.

Y17. Unable to Disable Reads/Writes to Performance Monitoring Related MSRs

Problem: The Performance Monitoring Available bit in the Miscellaneous Processor Features MSR (IA32_MISC_ENABLES.7) was defined so that when it is cleared to a 0, RDMSR/WRMSR/RDPMC instructions would return all zeros for reads of and prevent any writes to Performance Monitoring related MSRs. Currently it is possible to read from or write to Performance Monitoring related MSRs when the Performance Monitoring Available bit is cleared to a 0.

Implication: It is not possible to disallow reads and writes to the Performance Monitoring MSRs. Intel has not observed this erratum with any commercially available software or system.

Workaround: None.

Status: For the steppings affected, see the *Summary of Tables of Changes.*

Y18. Move to Control Register Instruction May Generate a Breakpoint Report

Problem: A move (MOV) to Control Register (CR) instruction where Control Register is CR0, CR3 or CR4 may generate a breakpoint report.

Implication: MOV to Control Register Instruction is not expected to generate a breakpoint report.

Workaround: Ignore breakpoint data from MOV to CR instruction.

Status: For the steppings affected, see the *Summary of Tables of Changes.*

Y19. REP MOVS/STOS Executing with Fast Strings Enabled and Crossing Page Boundaries with Inconsistent Memory Types may use an Incorrect Data Size or Lead to Memory-Ordering Violations

Problem: Under certain conditions as described in the Software Developers Manual section “Out-of-Order Stores For String Operations in Pentium 4, Intel Xeon, and P6 Family Processors” the processor performs REP MOVS or REP STOS as fast strings. Due to this erratum fast string REP MOVS/REP STOS instructions that cross page boundaries from WB/WC memory types to UC/WP/WT memory types, may start using an incorrect data size or may observe memory ordering violations.

Implication: Upon crossing the page boundary the following may occur, dependent on the new page memory type:

- UC the data size of each write will now always be 8 bytes, as opposed to the original data size.
- WP the data size of each write will now always be 8 bytes, as opposed to the original data size and there may be a memory ordering violation.
- WT there may be a memory ordering violation.

Workaround: Software should avoid crossing page boundaries from WB or WC memory type to UC, WP or WT memory type within a single REP MOVS or REP STOS instruction that will execute with fast strings enabled.

Status: For the steppings affected, see the *Summary Tables of Changes.*

Y20. The FXSAVE, STOS, or MOVS Instruction May Cause a Store Ordering Violation When Data Crosses a Page with a UC Memory Type

Problem: If the data from an FXSAVE, STOS, or MOVS instruction crosses a page boundary from WB to UC memory type and this instruction is immediately followed by a second instruction that also issues a store to memory, the final data stores from both instructions may occur in the wrong order.

Implication: The impact of this store ordering behavior may vary from normal software execution to potential software failure. Intel has not observed this erratum in commercially available software.

Workaround: FXSAVE, STOS, or MOVS data must not cross page boundary from WB to UC memory type.

Status: For the steppings affected, see the *Summary Tables of Changes.*

Y21. Machine Check Exception May Occur Due to Improper Line Eviction in the IFU

Problem: The processor is designed to signal an unrecoverable Machine Check Exception (MCE) as a consistency checking mechanism. Under a complex set of circumstances involving multiple speculative branches and memory accesses, there exists a one cycle long window in which the processor may signal a MCE in the Instruction Fetch Unit (IFU) because instructions previously decoded have been evicted from the IFU. The one cycle long window is opened when an opportunistic fetch receives a partial hit on a previously executed but not as yet completed store resident in the store buffer. The resulting partial hit erroneously causes the eviction of a line from the IFU at a time when the processor is expecting the line to still be present. If the MCE for this particular IFU event is disabled, execution will continue normally.

Implication: While this erratum may occur on a system with any number of processors, the probability of occurrence increases with the number of processors. If this erratum does occur, a machine check exception will result. Note systems that implement an operating system that does not enable the Machine Check Architecture will be completely unaffected by this erratum (e.g., Windows* 95 and Windows 98).

Workaround: It is possible for BIOS code to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary of Tables of Changes.*

Y22. POPF and POPFD Instructions That Set the Trap Flag Bit May Cause Unpredictable Processor Behavior

Problem: In some rare cases, POPF and POPFD instructions that set the Trap Flag (TF) bit in the EFLAGS register (causing the processor to enter Single-Step mode) may cause unpredictable processor behavior.

Implication: Single-Step operation is typically enabled during software debug activities, not during normal system operation.

Workaround: There is no workaround for Single-Step operation in commercially available software. For debug activities on custom software the POPF and POPFD instructions could be immediately followed by a NOP instruction to facilitate correct execution.

Status: For the steppings affected, see the *Summary of Tables of Changes.*

Y23. Performance Event Counter Returns Incorrect Value on L2_LINES_IN Event

Problem: The performance event counter returns an incorrect value on L2_LINES_IN event (EMON event #24H) when the L2 cache is disabled.

Implication: Due to this erratum, L2_LINES_IN performance event counter should not be monitored while the L2 cache is disabled. This erratum has no functional impact.

Workaround: Ignore L2_LINES_IN event when the L2 cache is disabled.

Status: For the steppings affected, see the *Summary of Tables of Changes.*

Y24. VM Bit Will Be Cleared on a Double Fault Handler

Problem: Following a task switch to a Double Fault Handler that was initiated while the processor was in virtual-8086 (VM86) mode, the VM bit will be incorrectly cleared in EFLAGS.

Implication: When the OS recovers from the double fault handler, the processor will no longer be in VM86 mode

Workaround: None

Status: For the steppings affected, see the *Summary of Tables of Changes.*

Y25. Code Fetch Matching Disabled Debug Register May Cause Debug Exception

Problem: The bits L0-3 and G0-3 enable breakpoints local to a task and global to all tasks, respectively. If one of these bits is set, a breakpoint is enabled, corresponding to the addresses in the debug registers DR0-DR3. If at least one of these breakpoints is enabled, any of these registers are *disabled* (i.e., L*n* and G*n* are 0), and RW*n* for the disabled register is 00 (indicating a breakpoint on instruction execution), normally an instruction fetch will not cause an instruction-breakpoint fault based on a match with the address in the disabled register(s). However, if the address in a disabled register matches the address of a code fetch which also results in a page fault, an instruction-breakpoint fault will occur.

Implication: While debugging software, extraneous instruction-breakpoint faults may be encountered if breakpoint registers are not cleared when they are disabled. Debug software which does not implement a code breakpoint handler will fail, if this occurs. If a handler is present, the fault will be serviced. Mixing data and code may exacerbate this problem by allowing disabled data breakpoint registers to break on an instruction fetch.

Workaround: The debug handler should clear breakpoint registers before they become disabled.

Status: For the steppings affected, see the *Summary of Tables of Changes*.

Y26. Upper Four PAT Entries Not Usable With Mode B or Mode C Paging

Problem: The Page Attribute Table (PAT) contains eight entries, which must all be initialized and considered when setting up memory types for the Pentium III processor. However, in Mode B or Mode C paging, the upper four entries do not function correctly for 4-Kbyte pages. Specifically, bit 7 of page table entries that translate addresses to 4-kbyte pages should be used as the upper bit of a 3-bit index to determine the PAT entry that specifies the memory type for the page. When Mode B (CR4.PSE = 1) and/or Mode C (CR4.PAE) are enabled, the processor forces this bit to zero when determining the memory type regardless of the value in the page table entry. The upper four entries of the PAT function correctly for
2-Mbyte and 4-Mbyte large pages (specified by bit 12 of the page directory entry for those translations).

Implication: Only the lower four PAT entries are useful for 4-KB translations when Mode B or C paging is used. In Mode A paging (4-Kbyte pages only), all eight entries may be used. All eight entries may be used for large pages in Mode B or C paging.

Workaround: None identified.

Status: For the steppings affected, see the *Summary of Tables of Changes*

Y27. SSE/SSE2 Streaming Store Resulting in a Self-Modifying Code (SMC) Event May Cause Unexpected Behavior

Problem: An SSE or SSE2 streaming store that results in a Self-Modifying Code (SMC) event may cause unexpected behavior. The SMC event occurs on a full address match of code contained in L1 cache.

Implication: Due to this erratum, any of the following events may occur:

1. A data access break point may be incorrectly reported on the instruction pointer (IP) just before the store instruction.
2. A non-cacheable store can appear twice on the external bus (the first time it will write only 8 bytes, the second time it will write the entire 16 bytes).

Intel has not observed this erratum with any commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the *Summary of Tables of Changes.*

Y28. Removed; See Erratum Y4

Y29. Removed; See Erratum Y5

Y30. Removed; See Erratum Y6

Y31. Page with PAT (Page Attribute Table) Set to USWC (Uncacheable Speculative Write Combine) While Associated MTRR (Memory Type Range Register) Is UC (Uncacheable) May Consolidate to UC

Problem: A page whose PAT memory type is USWC while the relevant MTRR memory type is UC, the consolidated memory type may be treated as UC (rather than WC as specified in *IA-32 Intel® Architecture Software Developer's Manual*).

Implication: When this erratum occurs, the memory page may be as UC (rather than WC). This may have a negative performance impact.

Workaround: None identified.

Status: For the steppings affected, see the *Summary of Tables of Changes.*

Y32. Under Certain Conditions LTR (Load Task Register) Instruction May Result in System Hang

Problem: An LTR instruction may result in a system hang if all the following conditions are met:

1. Invalid data selector of the TR (Task Register) resulting with either #GP (General Protection Fault) or #NP (Segment Not Present Fault).
2. GDT (Global Descriptor Table) is not 8-bytes aligned.
3. Data BP (breakpoint) is set on cache line containing the descriptor data. When this erratum occurs, the memory page may be as UC (rather than WC). This may have a negative performance impact.

Implication: This erratum may result in system hang if all conditions have been met. This erratum has not been observed in commercial operating systems or software. For performance reasons, GDT is typically aligned to 8-bytes

Workaround: Software should align GDT to 8-bytes.

Status: For the steppings affected, see the *Summary of Tables of Changes.*

Y33. Loading from Memory Type USWC (Uncacheable Speculative Write Combine) May Get Its Data Internally Forwarded from a Previous Pending Store

Problem: A load from memory type USWC may get its data internally forwarded from a pending store. As a result, the expected load may never be issued to the external bus.

Implication: When this erratum occurs, a USWC load request may be satisfied without being observed on the external bus. There are no known usage models where this behavior results in any negative side-effects

Workaround: Do not use memory type USWC for memory that has read side-effects.

Status: For the steppings affected, see the *Summary of Tables of Changes.*

Y34. FPU Operand Pointer May Not be Cleared Following FINIT/FNINIT

Problem: Initializing the floating point state with either FINIT or FNINT, may not clear the x87 FPU Operand (Data) Pointer Offset and the x87 FPU Operand (Data) Pointer Selector (both fields form the FPUDataPointer). Saving the floating point environment with FSTENV, FNSTENV, or floating point state with FSAVE, FNSAVE or FXSAVE before an intervening FP instruction may save uninitialized values for the FPUDataPointer.

Implication: When this erratum occurs, the values for FPUDataPointer in the saved floating point image or floating point environment structure may appear to be random values. Executing any non-control FP instruction with memory operand will initialize the FPUDataPointer. Intel has not observed this erratum with any commercially available software.

Workaround: After initialization, do not expect the FPUDataPointer in a floating point state or floating point environment saved memory image to be correct, until at least one non-control FP instruction with a memory operand has been executed.

Status: For the steppings affected, see the *Summary of Tables of Changes.*

Y35. FSTP (Floating Point Store) Instruction under Certain Conditions May Result In Erroneously Setting a Valid Bit on an FP (Floating Point) Stack Register

Problem: An FSTP instruction with a PDE/PTE (Page Directory Entry/Page Table Entry) A/D bit update followed by user mode access fault due to a code fetch to a page that has supervisor only access permission may result in erroneously setting a valid bit of an FP stack register. The FP top of stack pointer is unchanged.

Implication: This erratum may cause an unexpected stack overflow.

Workaround: User mode code should not count on being able to recover from illegal accesses to memory regions protected with supervisor only access when using FP instructions.

Status: For the steppings affected, see the *Summary of Tables of Changes.*

Y36. Snoops during the Execution of a HLT (Halt) Instruction May Lead to Unpredictable System Behavior

Problem: If during the execution of a HLT instruction an external snoop causes an eviction from the instruction fetch unit (IFU) instruction cache, the processor may, on exit from the HLT state, erroneously read stale data from the victim cache.

Implication: This erratum may lead to unpredictable system behavior. Intel has only observed this condition in non-mobile configurations.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary of Tables of Changes.*

Y37. Invalid Entries in Page-Directory-Pointer-Table Register (PDPTR) May Cause General Protection (#GP) Exception If the Reserved Bits Are Set to One

Problem: Invalid entries in the Page-Directory-Pointer-Table Register (PDPTR) that have the reserved bits set to one may cause a General Protection (#GP) exception.

Implication: Intel has not observed this erratum with any commercially available software.

Workaround: Do not set the reserved bits to one when PDPTR entries are invalid.

Status: For the steppings affected, see the *Summary of Tables of Changes.*

Y38. INIT Does Not Clear Global Entries in the TLB

Problem: INIT may not flush a TLB entry when:

1. The processor is in protected mode with paging enabled and the page global enable flag is set (PGE bit of CR4 register)
2. G bit for the page table entry is set
3. TLB entry is present in TLB when INIT occurs

Implication: Software may encounter unexpected page fault or incorrect address translation due to a TLB entry erroneously left in TLB after INIT.

Workaround: Write to CR3, CR4 (setting bits PSE, PGE or PAE) or CR0 (setting bits PG or PE) registers before writing to memory early in BIOS code to clear all the global entries from TLB.

Status: For the steppings affected, see the *Summary of Tables of Changes.*

Y39. Use of Memory Aliasing with Inconsistent Memory Type May Cause System Hang

Problem: Software that implements memory aliasing by having more than one linear addresses mapped to the same physical page with different cache types may cause the system to hang. This would occur if one of the addresses is non-cacheable used in code segment and the other a cacheable address. If the cacheable address finds its way in instruction cache, and non-cacheable address is fetched in IFU, the processor may invalidate the non-cacheable address from the fetch unit. Any micro-architectural event that causes instruction restart will expect this instruction to still be in fetch unit and lack of it will cause system hang.

Implication: This erratum has not been observed with commercially available software.

Workaround: Although it is possible to have a single physical page mapped by two different linear addresses with different memory types, Intel has strongly discouraged this practice as it may lead to undefined results. Software that needs to implement memory aliasing should manage the memory type consistency.

Status: For the steppings affected, see the *Summary of Tables of Changes.*

Y40. Machine Check Exception May Occur When Interleaving Code between Different Memory Types

Problem: A small window of opportunity exists where code fetches interleaved between different memory types may cause a machine check exception. A complex set of micro-architectural boundary conditions is required to expose this window.

Implication: Interleaved instruction fetches between different memory types may result in a machine check exception. The system may hang if machine check exceptions are disabled. Intel has not observed the occurrence of this erratum while running commercially available applications or operating systems.

Workaround: Software can avoid this erratum by placing a serializing instruction between code fetches between different memory types.

Status: For the steppings affected, see the *Summary of Tables of Changes.*

Y41. Split I/O Writes Adjacent to Retry of APIC End of Interrupt (EOI) Request May Cause Livelock Condition

Problem: When Split I/O instruction writes occur adjacent to a retry of a Local APIC End of Interrupt (EOI) request by the chipset, a livelock condition may result. The required sequences of events are:

4. The processor issues a Local APIC EOI message.
5. The chipset responds with a retry because its downstream ports are full. It expects the processor to return with the same EOI request.
6. The processor issues a Split I/O write instruction instead.
7. The chipset responds with a retry because it expected the APIC EOI.
8. The processor insists the Split I/O write instruction must be completed and issues write instruction again.

Implication: A processor livelock may occur causing a system hang. This issue has only been observed in synthetic lab testing conditions and has not been seen in any commercially available applications. The erratum does not occur with Intel mobile chipset-based platforms.

Workaround: Use the PIC instead of the APIC for the interrupt controller.

Status: For the steppings affected, see the *Summary of Tables of Changes.*

Y42. General Protection (#GP) Fault May Not Be Signaled On Data Segment Limit Violation above 4-G Limit

Problem: Memory accesses to flat data segments (base = 00000000h) that occur above the 4-G limit (0ffffffffh) may not signal a #GP fault.

Implication: When such memory accesses occur, the system may not issue a #GP fault.

Workaround: Software should ensure that memory accesses do not occur above the 4-G limit (0ffffffffh).

Status: For the steppings affected, see the *Summary of Tables of Changes.*

Y43. DR3 Address Match on MOVD/MOVQ/MOVNTQ Memory Store Instruction May Incorrectly Increment Performance Monitoring Count for Saturating SIMD Instructions Retired (Event CFH)

Problem: Performance monitoring for Event CFH normally increments on saturating SIMD instruction retired. Regardless of DR7 programming, if the linear address of a retiring memory store MOVD/MOVQ/MOVNTQ instruction executed matches the address in DR3, the CFH counter may be incorrectly incremented.

Implication: The value observed for performance monitoring count for saturating SIMD instructions retired may be too high. The size of the error is dependent on the number of occurrences of the conditions described above, while the counter is active.

Workaround: None identified.

Status: For the steppings affected, see the *Summary of Tables of Changes.*

Y44. Processor INIT# Will Cause a System Hang if Triggered during an NMI Interrupt Routine Performed during Shutdown

Problem: During the execution of an NMI interrupt handler, if shutdown occurs followed by the INIT# signal being triggered, the processor will attempt initialization but fail soft reset.

Implication: Due to this erratum, the system may hang.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Y45. Certain Performance Monitoring Counters Related to Bus, L2 Cache and Power Management are Inaccurate

Problem: All Performance Monitoring Counters in the ranges 21H-3DH and 60H-7FH may have inaccurate results up to ±7.

Implication: There may be a small error in the affected counts.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Y46. CS Limit Violation on RSM May be Serviced before Higher Priority Interrupts/Exceptions

Problem: When the processor encounters a CS (Code Segment) limit violation, a #GP (General Protection Exception) fault is generated after all higher priority Interrupts and exceptions are serviced. Because of this erratum, if RSM (Resume from System Management Mode) returns to execution flow where a CS limit violation occurs, the #GP fault may be serviced before a higher priority Interrupt or Exception (e.g. NMI (Non-Maskable Interrupt), Debug break(#DB), Machine Check (#MC), etc.)

Implication: Operating systems may observe a #GP fault being serviced before higher priority Interrupts and Exceptions.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Y47. A Write to an APIC Register Sometimes May Appear to Have Not Occurred

Problem: With respect to the retirement of instructions, stores to the uncacheable memory-based APIC register space are handled in a non-synchronized way. For example if an instruction that masks the interrupt flag, e.g. CLI, is executed soon after an uncacheable write to the Task Priority Register (TPR) that lowers the APIC priority, the interrupt masking operation may take effect before the actual priority has been lowered. This may cause interrupts whose priority is lower than the initial TPR, but higher than the final TPR, to not be serviced until the interrupt enabled flag is finally set, i.e. by STI instruction. Interrupts will remain pending and are not lost.

Implication: In this example the processor may allow interrupts to be accepted but may delay their service.

Workaround: This non-synchronization can be avoided by issuing an APIC register read after the APIC register write. This will force the store to the APIC register before any subsequent instructions are executed. No commercial operating system is known to be impacted by this erratum.

Status: For the steppings affected, see the *Summary Tables of Changes.*

Y48. The Processor May Report a #TS Instead of a #GP Fault

Problem: A jump to a busy TSS (Task-State Segment) may cause a #TS (invalid TSS exception) instead of a #GP fault (general protection exception).

Implication: Operation systems that access a busy TSS may get invalid TSS fault instead of a #GP fault. Intel has not observed this erratum with any commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Y49. Writing the Local Vector Table (LVT) when an interrupt is pending may cause an unexpected interrupt

Problem: If a local interrupt is pending when the LVT entry is written, an interrupt may be taken on the new interrupt vector even if the mask bit is set.

Implication: An interrupt may immediately be generated with the new vector when a LVT entry is written, even if the new LVT entry has the mask bit set. If there is no Interrupt Service Routine (ISR) set up for that vector the system will GP fault. If the ISR does not do an End of Interrupt (EOI) the bit for the vector will be left set in the in-service register and mask all interrupts at the same or lower priority.

Workaround: Any vector programmed into an LVT entry must have an ISR associated with it, even if that vector was programmed as masked. This ISR routine must do an EOI to clear any unexpected interrupts that may occur. The ISR associated with the spurious vector does not generate an EOI, therefore the spurious vector should not be used when writing the LVT.

Status: For the steppings affected , see the *Summary Tables of Changes*.

Y50. Using 2M/4M Pages When A20M# is Asserted May Result in Incorrect Address Translations

Problem: An external A20M# pin if enabled forces address bit 20 to be masked (forced to zero) to emulates real-address mode address wraparound at 1 megabyte. However, if all of the following conditions are met, address bit 20 may not be masked.

- paging is enabled
- a linear address has bit 20 set
- the address references a large page
- A20M# is enabled

Implication: When A20M# is enabled and an address references a large page the resulting translated physical address may be incorrect. This erratum has not been observed with any commercially available operating system.

Workaround: Operating systems should not allow A20M# to be enabled if the masking of address bit 20 could be applied to an address that references a large page. A20M# is normally only used with the first megabyte of memory.

Status: For the steppings affected, see the *Summary Tables of Changes.*

Y51. Premature Execution of a Load Operation Prior to Exception Handler Invocation

Problem: If any of the below circumstances occur it is possible that the load portion of the instruction will have executed before the exception handler is entered.

1) If an instruction that performs a memory load causes a code segment limit violation

2) If a waiting floating-point instruction or MMX instruction that performs a memory load has a floating-point exception pending

3) If an MMX or SSE instruction that performs a memory load and has either CR0.EM=1 (Emulation bit set), or a floating-point Top-of-Stack (FP TOS) not equal to 0, or a DNA exception pending

Implication: In normal code execution where the target of the load operation is to write back memory there is no impact from the load being prematurely executed, nor from the restart and subsequent re-execution of that instruction by the exception handler. If the target of the load is to uncached memory that has a system side-effect, restarting the instruction may cause unexpected system behavior due to the repetition of the side-effect.

Workaround: Code which performs loads from memory that has side-effects can effectively workaround this behavior by using simple integer-based load instructions when accessing side-effect memory and by ensuring that all code is written such that a code segment limit violation cannot occur as a part of reading from side-effect memory.

Status: For the steppings affected, see the *Summary Tables of Changes.*

Y52. Incorrect Address Computed For Last Byte of FXSAVE/FXRSTOR Image Leads to Partial Memory Update

Problem: A partial memory state save of the 512-byte FXSAVE image or a partial memory state restore of the FXRSTOR image may occur if a memory address exceeds the 64KB limit while the processor is operating in 16-bit mode or if a memory address exceeds the 4GB limit while the processor is operating in 32-bit mode.

Implication: FXSAVE/FXRSTOR will incur a #GP fault due to the memory limit violation as expected but the memory state may be only partially saved or restored.

Workaround: Software should avoid memory accesses that wrap around the respective 16-bit and 32-bit mode memory limits.

Status: For the steppings affected, see the *Summary Tables of Changes.*

Y53. Values for LBR/BTS/BTM will be Incorrect after an Exit from SMM

Problem: After a return from SMM (System Management Mode), the CPU will incorrectly update the LBR (Last Branch Record) and the BTS (Branch Trace Store), hence rendering their data invalid. The corresponding data if sent out as a BTM on the system bus will also be incorrect.
Note: This issue would only occur when one of the 3 above mentioned debug support facilities are used.

Implication: The value of the LBR, BTS, and BTM immediately after an RSM operation should not be used.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes.*

Y54. FP Inexact-Result Exception Flag May Not Be Set

Problem: When the result of a floating-point operation is not exactly representable in the destination format (1/3 in binary form, for example), an inexact-result (precision) exception occurs. When this occurs, the PE bit (bit 5 of the FPU status word) is normally set by the processor. Under certain rare conditions, this bit

may not be set when this rounding occurs. However, other actions taken by the processor (invoking the software exception handler if the exception is unmasked) are not affected. This erratum can only occur if the floating-point operation which causes the precision exception is immediately followed by one of the following instructions:

- FST m32real
- FST m64real
- FSTP m32real
- FSTP m64real
- FSTP m80real|
- FIST m16int
- FIST m32int
- FISTP m16int|
- FISTP m32int
- FISTP m64int

Note that even if this combination of instructions is encountered, there is also a dependency on the internal pipelining and execution state of both instructions in the processor.

Implication: Inexact-result exceptions are commonly masked or ignored by applications, as it happens frequently, and produces a rounded result acceptable to most applications. The PE bit of the FPU status word may not always be set upon receiving an inexact-result exception. Thus, if these exceptions are unmasked, a floating-point error exception handler may not recognize that a precision exception occurred. Note that this is a "sticky" bit, i.e., once set by an inexact-result condition, it remains set until cleared by software.

Workaround: This condition can be avoided by inserting two non-floating-point instructions between the two floating-point instructions.

Status: For the steppings affected, see the *Summary Tables of Changes.*

Y55. MOV To/From Debug Registers Causes Debug Exception

Problem: When in V86 mode, if a MOV instruction is executed to/from debug register, a general-protection exception (#GP) should be generated. However, in the case when the general detect enable flag (GD) bit is set, the observed behavior is that a debug exception (#DB) is generated instead.

Implication: With debug-register protection enabled (i.e., the GD bit set), when attempting to execute a MOV on debug registers in V86 mode, a debug exception will be generated instead of the expected general-protection fault.

Workaround: In general, operating systems do not set the GD bit when they are in V86 mode. The GD bit is generally set and used by debuggers. The debug exception handler should check that the exception did not occur in V86 mode before continuing. If the exception did occur in V86 mode, the exception may be directed to the general-protection exception handler.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Y56. SYSENTER/SYSEXIT Instructions Can Implicitly Load "Null Segment Selector" to SS and CS Registers

Problem: According to the processor specification, attempting to load a null segment selector into the CS and SS segment registers should generate a General Protection Fault (#GP). Although loading a null segment selector to the other segment registers is allowed, the processor will generate an exception when the segment register holding a null selector is used to access memory. However, the SYSENTER instruction can implicitly load a null value to the SS segment selector. This can occur if the value in SYSENTER_CS_MSR is between FFF8h and FFFBh when the SYSENTER instruction is executed. This behavior is part of the SYSENTER/SYSEXIT instruction definition; the content of the SYSTEM_CS_MSR is always incremented by 8 before it is loaded into the SS. This operation will set the null bit in the segment selector if a null result is generated, but it does not generate a #GP on the SYSENTER instruction itself. An exception will be generated as expected when the SS register is used to access memory, however. The SYSEXIT instruction will also exhibit this behavior for both CS and SS when executed with the value in SYSENTER_CS_MSR between FFF0h and FFF3h, or between FFE8h and FFEBh, inclusive.

Implication: These instructions are intended for operating system use. If this erratum occurs (and the OS does not ensure that the processor never has a null segment selector in the SS or CS segment registers), the processor's behavior may become unpredictable, possibly resulting in system failure.

Workaround: Do not initialize the SYSTEM_CS_MSR with the values between FFF8h and FFFBh, FFF0h and FFF3h, or FFE8h and FFEBh before executing SYSENTER or SYSEXIT.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Y57. The BS Flag in DR6 May be Set for Non-Single-Step #DB Exception

Problem: DR6 BS (Single Step, bit 14) flag may be incorrectly set when the TF (Trap Flag, bit 8) of the EFLAGS Register is set, and a #DB (Debug Exception) occurs due to one of the following:

- DR7 GD (General Detect, bit 13) being bit set;
- INT1 instruction;
- Code breakpoint

Implication: The BS flag may be incorrectly set for non-single-step #DB exception.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes.*

§

Specification Changes

There are no specification changes in this Specification Update revision.

§

Specification Clarifications

There are no specification clarifications in this Specification Update revision.

§

Documentation Changes

There are no documentation changes in this Specification Update revision.

Note: Documentation changes for Intel® 64 and IA-32 Arcitecurtes Software Developer Manual volumes 1, 2A, 2B, 3A, and 3B will be posted in a separate document named *Intel® 64 and IA-32 Architecture Software Developer's Manual Documentation Changes*. Follow the link below to become familiar with this file.

http://developer.intel.com/design/pentium4/specupdt/252046.htm